



Frühjahrsfachgespräch 2005

Pluggable Authentication Modules

Florian Brand 2005-02-23

Agenda:

- Unit 1: Einführung
 - Grundlagen
 - Unix Authentifikation
 - PAM Konfiguration
- Unit 2: Passwörter
 - Passwort Sicherheit
 - Netzwerk Authentifikation
- Unit 3: Userbasierte Konfiguration
 - Mechanismen zum Sperren von Usern
 - Anwendung

Einführung

Agenda

- Grundlagen der Authentifizierung
- Unix Authentifizierung
- LIBC Funktionen
- Name Service Switch
- PAM Grundlagen
- Grundkonfiguration

Besprochene PAM Module

- pam_securetty: Einschränkung von root auf sichere Terminals
- pam_unix: Standard Authentifizierung nach Unix-Art
- pam_krb5: Kerberos Authentifizierung
- pam_session Sonderrechte für normale User
- pam_stack: Verschachtelung von PAM Aufrufen

Grundlagen der Authentifizierung

- Vier Teilbereiche
 - Authentifizierung
 - Autorisierung
 - Aktualisierung von Passwörtern
 - Management der Session
- Verschiedene Applikationen
 - login
 - sshd
 - gdm
 - ...

User und Gruppen

- Jeder User Account hat eine eindeutige UID
- Jeder Gruppen Account hat eine eindeutige GID
- Jeder Prozess hat einen Security Context: Die *Persona*
 - UID, primäre GID, zusätzliche GIDs
 - eventuell eine effektive UID und GID (setUID Prozesse)

Standard C Bibliothek

- Stellt eine Standard API für Systemfunktionen zur Verfügung
 - Wird von fast jedem Programm verwendet
 - Ermöglicht Systemcalls
 - Namensauflösung, Speichermanagement, I/O, Sockets, Prozessverwaltung,...
- Prozesse können Userinformationen über libc-Funktionen bekommen

Unix Authentifikation

- Benutzer gibt Username und Passwort ein
- Applikation ruft die Funktion `getpwnam()` der LIBC auf, um die gespeicherten Informationen des Users zu bekommen
- Die Applikation hasht das eingegebene Passwort und vergleicht es mit dem gespeicherten Hash
- Wenn der Hash übereinstimmt, ist der User authentifiziert

Name Service der LIBC

- `getpwnam()` und andere Funktionen verwenden den Name Service um einen Namen auf eine Information zu mappen
- Ursprünglich wurden nur die lokalen Dateien `/etc/passwd` und `/etc/hosts` verwendet
- Um weitere Dienste hinzuzufügen musste früher die LIBC umgeschrieben werden.

Name Service Switch

- mittels NSS können einfach neue Namensdienste ohne Neukompilation der LIBC hinzugefügt werden
- Neue Dienste müssen unter `/lib/libnss_<service>.so` installiert werden
- Um den neuen Dienst zu verwenden muss die Datei `/etc/nsswitch.conf` angepasst werden

/etc/nsswitch.conf

- Erstes Element der Zeile gibt den Typ der Datenquelle an:
 - aliases, ethers, group, hosts, netgroup, networks, passwd, protocols, rpc, services, passwd, shadow
- Der Rest der Zeile gibt die verwendeten Namensdienste an
 - Die Dienste werden in der angegebenen Reihenfolge abgefragt
 - z.B: passwd: files nis ldap

Ergebnis der Namensauflösung

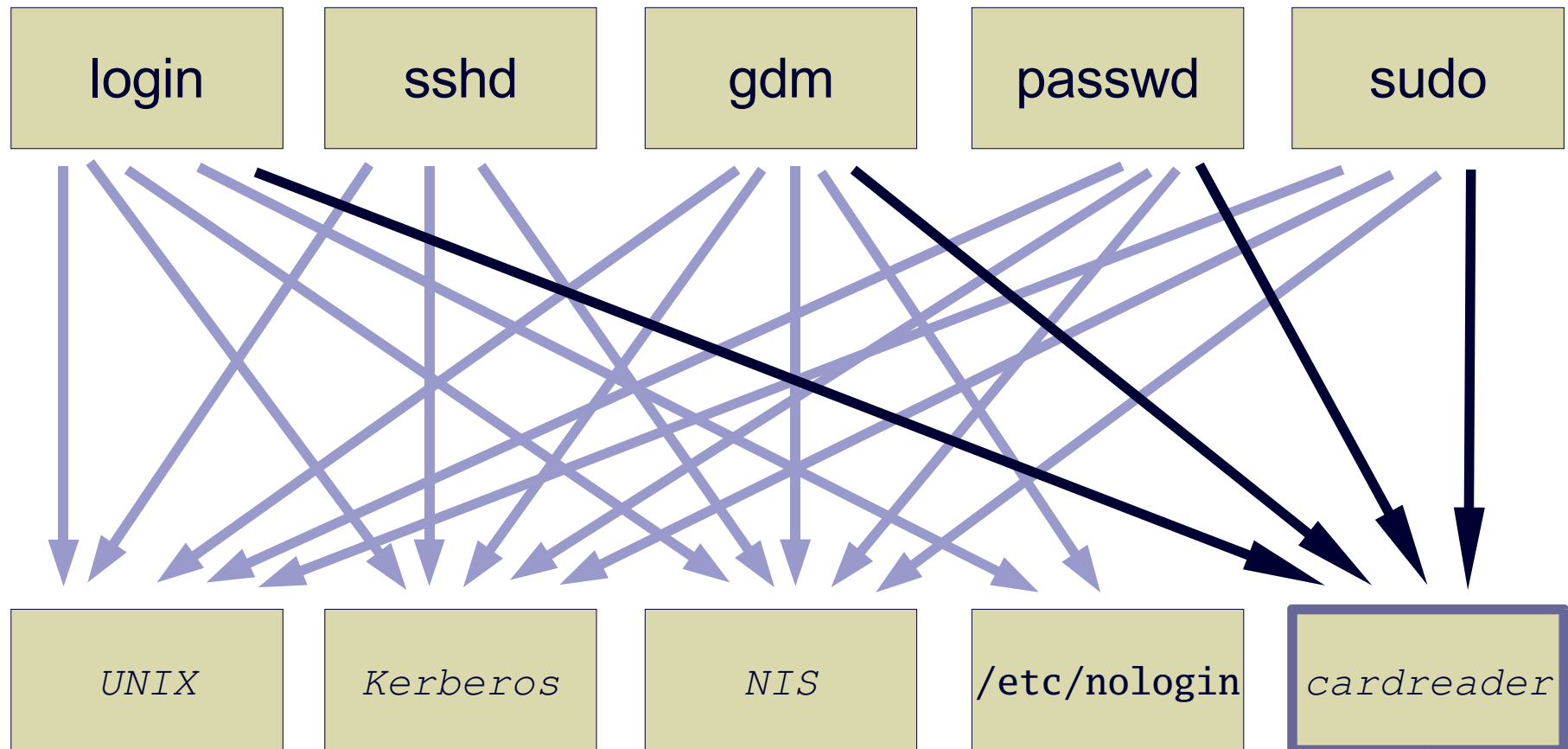
- Vier mögliche Rückgabewerte:
 - SUCCESS Service OK, Name gefunden
 - NOTFOUND Service OK, Name nicht gefunden
 - UNAVAIL Service nicht verfügbar
 - TRYAGAIN Dienst temporär nicht verfügbar
- Es ist möglich, auf diese Rückgabewerte zu reagieren
 - networks: nis [NOTFOUND=return] files
 - Default: [SUCCEES=return !SUCCESS=continue]

getent

- **getent database**
 - listet alle Objekte in der angegebenen Datenbank
 - z.B: **getent services**
- **getent database name**
 - Gibt nur das angegebene Objekt aus
 - z.B: **getent passwd smith**

Authentifizierung früher...

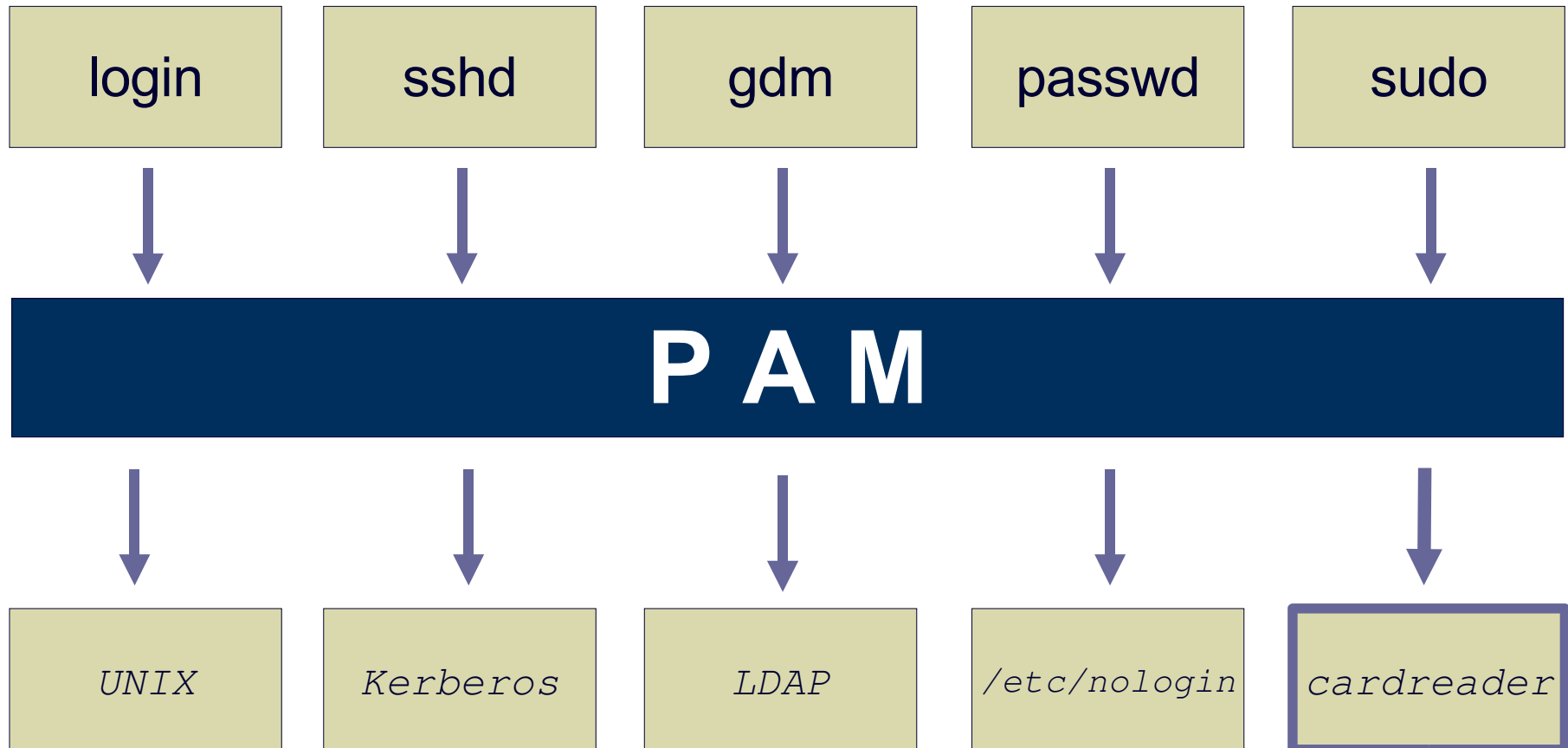
Applikationen



Dienste

Authentifizierung heute

Applikationen



Dienste

Was ist PAM ?

- PAM = Pluggable Authentication Modules
- Zentrale Bibliotheken für die Authentifikation
- Konfiguration mittels Textdateien
- Modularer Aufbau für einfache Anpassung

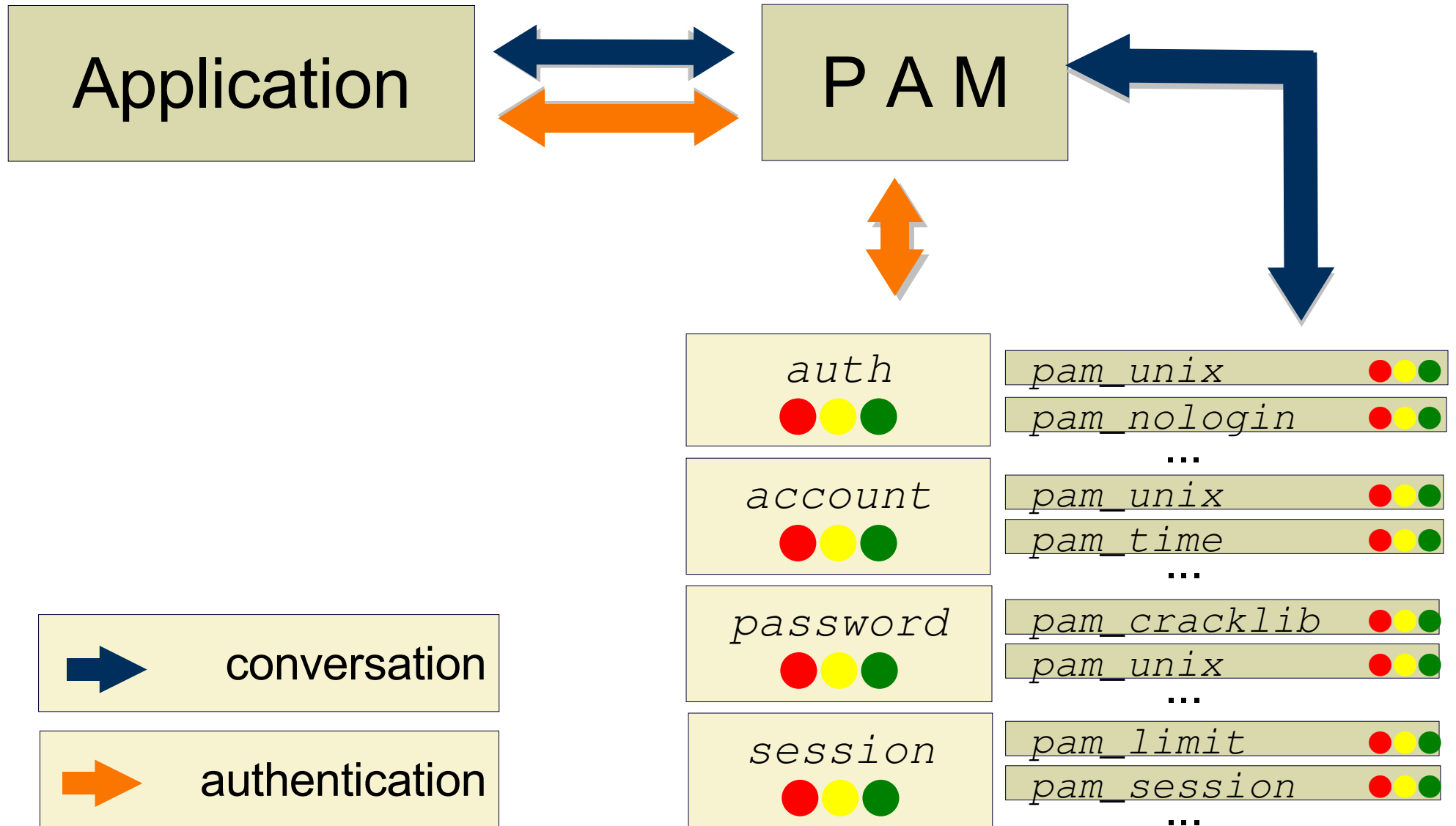
PAM Vorteile

- Zentrale Bibliothek
 - eine Code-Basis, daher einfach wartbar
 - Applikationsprogrammierer müssen sich nicht mit den Details der Authentifizierung beschäftigen
- Konfigurierbare Module
 - Anpassung der Authentifizierung ohne Neucompilation der Anwendung
 - Einfaches Einbetten neuer Module

Ist PAM ein Standard?

- PAM ist in RFC 86.0 beschrieben
 - 1995 von Vipin Smar und Roland Schemers entwickelt
 - Viele Applikationen benutzen PAM
 - Red Hat verwendet PAM seit Red Hat Linux 4.2
- Aber:
 - PAM wird nur von Applikationen verwendet, die gegen PAM gelinkt sind
 - Nicht alle Betriebssysteme unterstützen PAM (z.B. AIX)

Funktionsweise von PAM



Konfiguration

- PAM wird über zwei Ebenen konfiguriert:
 - **/etc/pam.d/*** bzw. **/etc/pam.conf** spezifizieren die verwendeten Module
 - Konfiguration der Module in **/etc/security/*** und anderen Dateien
- Änderungen werden sofort aktiv.
 - daher sofort auf einem zweiten Terminal testen

Einfacher Syntax:

- **<Typ> <Aktion> <Bibliothek> [Optionen]**
- Typ gibt den Authentifikationstyp an:
 - auth: Authentifizierung
 - account: Authorisation
 - password Aktualisierung von Passwörter
 - session Initialisierung der Session des Users
- Die Aktion gibt an wie PAM auf Rückgabe werte der einzelnen Module reagiert
- Beispiel:
auth required pam_unix.so nullok

Einfacher Syntax: Aktionen

- required: Bei Erfolg weiterprüfen
- requisite: Wie required, aber sofortiger Abbruch bei Fehlschlag
- optional: Ergebnis ignorieren
- sufficient: Bei Erfolg sofortige Rückkehr, sonst ignorieren

Beispiel

/etc/pam.d/login:

auth	requisite	/lib/security/pam_securetty.so
auth	sufficient	/lib/security/pam_unix.so nullok
auth	required	/lib/security/pam_ldap.so \ use_first_pass
account	required	/lib/security/pam_unix.so
password	required	/lib/security/pam_cracklib.so
password	required	/lib/security/pam_unix.so nullok \ shadow md5
session	required	/lib/security/pam_unix.so
session	required	/lib/security/pam_limits.so
session	optional	/lib/security/pam_console.so

Konfiguration II

- Separate Konfigurationsdateien für jeden einzelnen Dienst
- User, die in einem Dienst gesperrt sind, könnten sich über einen anderen Dienst einloggen
- Mittels pam_stack können alle Dienste gemeinsam konfiguriert werden

Beispiel

/etc/pam.d/login:

```
auth  required      /lib/security/pam_securetty.so
auth  required      /lib/security/pam_stack.so \
service=system-auth
```

/etc/pam.d/system-auth:

```
auth  required      /lib/security/pam_env.so
auth  sufficient     /lib/security/pam_unix.so \
likeauth nullok
auth  sufficient     /lib/security/pam_ldap.so \
use_first_pass
auth  required      /lib/security/pam_deny.so
```

```
auth  required      /lib/security/pam_nologin.so
```

Erweiterter Syntax

- Konfiguration, wie PAM auf bestimmte Return-Codes reagiert
 - Damit ist ein Fine-Tuning möglich
- Weist dem Rückgabewert eine Aktion zu
 - ca. 30 verschiedene Rückgabewerte
- Sechs Standard Aktionen
 - ok, done, bad, die, ignore, reset

Aktionen

- ok: Erfolg, weiter prüfen
- done: Erfolg, zurück zur Applikation
- bad: Fehlschlag, weiter prüfen
- die: Fehlschlag, zurück zur Applikation
- ignore: Ergebnis ignorieren, weiter prüfen
- reset: Alle vorherigen Ergebnisse ignorieren, weiter prüfen

Rückgabewerte:

- success: Test erfolgreich
- new_authtok_reqd: Passwort abgelaufen, Neueingabe wenn möglich
- ignore: Dieses Modul ignorieren
- default: Default Aktion, wenn kein anderer Rückgabewert zutrifft
- user_unknown: Unbekannter User
- authinfo_unavail: Dienst nicht verfügbar

Verhältnis zur Standard Syntax

- Standard Syntax kann mit der erweiterten Syntax abgebildet werden
- required entspricht:
[success=ok new_authtok_reqd=ok ignore=ignore default=bad]
- sufficient entspricht:
[success=done new_authtok_reqd=done default=ignore]

Unit 2

Passwörter

Agenda

- Security Policy
- Elemente der Passwortsicherheit
- Passwort-Komplexität
- Passwort Aging
- Account Sperre bei Fehleingaben
- Netzwerk basierte Authentifikation
 - NIS
 - LDAP
 - Kerberos

Besprochene PAM Module

- pam_cracklib: Modul zur Kontrolle der Passwortqualität
- pam_unix: Allgemeines Authentifizierungsmodul, Speziell: Passworthistorie, Passwort-Aging
- pam_tally: Automatische Sperre von Benutzerkonten bei häufiger Passwortfehleingabe

Passwortsicherheit: pam_cracklib

- Gibt an viele Credits ein gutes Passwort benötigt
- 1 Credit = 1 Character
- Extracredits:
 - Für bestimmte Credits kann der User Extra credits bekommen
 - Anzahl der Extracredits begrenzbare
 - Arten von Extracredits:
 - ucredit Großbuchstaben
 - lcredit Kleinbuchstaben
 - dcredit Ziffern
 - ocredit Sonderzeichen

Password Aging: pam_unix

- Feature der Shadow Passwörter
- Konfiguration über chage
- Mögliche Parameter:
 - Maximale Gültigkeit eines Passworts
 - Minimale Gültigkeit eines Passworts
 - Datum der letzten Passwortänderung
 - Warnperiode
 - Ablaufdatum des Accounts

Passwordhistory: pam_unix

- Beispiel:
passwordrequired pam_unix.so remember=5
- Alte Passwort-Hashes können gespeichert werden
- Vorteil: User können alte Passwörter nicht wieder verwenden
- Da Hashes sich bei bereits einem Zeichen Unterschied ändern, ist der Vorteil gering
- pam_cracklib überprüft ebenfalls das letzte Passwort

Account-Sperren: pam_tally

- Aussperren bei zu häufiger Passwortfehleingabe
- Zwei Teile: pam_tally.so und pam_tally
- Zwei Aufrufe von pam_tally notwendig:
 - pam_tally.so erhöht Fehlschlagscounter in der Authentifikationsphase
 - pam_tally.so überprüft den Fehlschlagscounter in der Autorisierungsphase
- Sonderbehandlung für Root
- Zwei wichtige Parameter:
 - even_deny_root: Sperrt auch den Rootaccount, Vorsicht DOS Möglichkeit!
 - deny=n Anzahl der Fehlversuche
- pam_tally manipuliert /var/log/faillog

Beispiel

/etc/pam.d/system-auth:

password required	/lib/security/pam_cracklib.so \ minlength=20 ucredit=5 lcredit=5 \ dcredit=5 ocredit=5 diffok=5 \ retry=3
password required	/lib/security/pam_unix.so \ nullok remember=5

Netzwerkauthentifikation

- Zentrale Verwaltung der User Accounts
- verschiedene Quellen für Usernamen und Credentials nutzbar
- Zwei Mechanismen:
 - Einbindung in NSS (pam_unix)
 - spezifische PAM Module
- PAM Support notwendig, damit Passwortänderungen vorgenommen werden können

PAM und NSS

- NSS nimmt Namensauflösungen vor
 - z.B: Name->UID
- PAM authentifiziert User
 - Weitere Accountdaten normalerweise irrelevant
- Verzeichnisdienste (NIS, LDAP)
 - haben vollständige Informationen
 - können von PAM und NSS verwendet werden
- Authentifikationsdienste (Kerberos, SMB)
 - kennen nur Username und "Credentials"
 - nur von PAM verwendbar
 - weitere Eigenschaften müssen von einem Verzeichnisdienst aufgelöst werden

NIS

- Einbindung über NSS
- Verzeichnisdienst
- PAM Konfiguration für Passwortänderungen
- Unverschlüsselte Übertragung von Passwortdaten
- Kombination mit Kerberos für eine sichere Authentifizierung

Kerberos

- Reiner Authentifizierungsdienst
- Initiale Kommunikation mit dem Kerberos Server
- Erfolgreiche Entschlüsselung des Ticket Granting Ticket == Erfolg
- Ticket Authentifizierung an weiteren Diensten wird nicht über PAM ausgeführt

LDAP

- Verzeichnisdienst
- Wird sowohl in NSS als auch PAM integriert
- pam_ldap und nss_ldap werden unter /etc/ldap.conf konfiguriert
- LDAP Security Policies müssen für Passwortänderungen angepasst werden.
- pam_ldap wird nur bei LDAP Authentifikation benötigt.
- Erweiterter Syntax nötig um bei LDAP Server Ausfall noch lokale Logins zu ermöglichen:
account [default=bad user_unknown=ignore \
success=ok authinfo_unavail=ignore] \
pam_ldap.so

SMB

- Verwendung als Authentifizierungsdienst
- pam_smb
 - benötigt lokale Accounts
- pam_winbind
 - generiert UIDs
 - kann über LDAP verteilt werden
- ADS kann um Posix Accounts erweitert werden

Home-Verzeichnisse

- `pam_mkhomedir` generiert Homeverzeichnisse beim ersten Einloggen
- “Unix Musterlösung”: Kombination mit NFS und Automounter

Unit 3

Userbasierte Konfiguration

Agenda

- Sperrung von Einzeldiensten
- Zeitbasierte Zugangskontrolle
- Terminal/IP basierte Zugangskontrolle
- Einschränkung von System Ressourcen

Besprochene PAM Module

- `pam_listfile:` Einfache Zugangskontrolllisten
- `pam_localuser:` Freigabe lokaler User
- `pam_time:` Zeitbasierte Zugangskontrolle
- `pam_access:` Zugangskontrolle auf Basis von Terminals oder IP Adressen von denen ein User zugreifen darf.
- `pam_limits:` Einschränkung von Systemressourcen

Grundlagen

- Security Policy: Freischaltung von Usern nur für bestimmte Dienste
- Mögliche Kriterien
 - Userliste
 - Art der Authentifizierung: Lokal, Netzwerk
 - Location des Users: (Terminal, IP)
 - Uhrzeit
 - ...
- Sperren von Usern in der Authorisationsphase
 - Manche Dienste authentifizieren den User selbst (Beispiel SSH)

Userlisten: pam_listfile

- Syntax:

```
account    required    pam_listfile.so file=<FILE> \  
            item=<ITEM> sense=<allow|deny>\  
            onerr=<fail|succeed>
```

- Item: <tty|user|group|rhost|ruser|shell>

- Beispiel:

```
account    required    pam_listfile.so item=user \  
            file=/etc/goodusers \  
            sense=allow onerr=fail
```

Bedingungen: pam_succeed_if

■ Arithmetische Vergleiche möglich

- Größer/Kleiner: $>$, $>=$, $..$
- Gleichheit: eq , ne , $=$, $!=$
- Gruppenzugehörigkeit: $ingroup$, $notingroup$
- Wildcard Match $=\sim$, $!\sim$

■ Beispiele:

```
account sufficient pam_succeed_if.so uid < 100 quiet
account sufficient pam_succeed_if.so shell !~ nologin
```

Lokale User: pam_localuser

- Erlaubt allen in /etc/passwd gelisteten Usern Zugriff
- Alternative Passwortdatei mit der File Option
- Beispiel:

/etc/pam.d/system-auth:

account sufficient
account sufficient

account required

pam_localuser.so
**pam_listfile.so **
item=user
**file=/etc/goodusers **
**sense=allow **
onerr=fail
pam_deny.so

Location: pam_access

- Konfiguration: **/etc/security/access.conf**
- Syntax **<+ | -> : <users> : <origin>**
 - + allow, - deny
 - <users>: Leerzeichen separierte Liste von Usern, Gruppen, Netzgruppen
 - <origin>: Liste von TTYs, IP Adressen
- Vorsicht: Per Default Allow!
- Beispiel:
+ : root : tty1
+ : ALL EXCEPT root : ALL
- : ALL : ALL

Uhrzeit: pam_time

- Konfiguration: **/etc/security/time.conf**
- Syntax: **<services>; <ttys>; <users>; <times>**
 - Wildcards möglich
 - <services>: PAM Dienst
 - <ttys>: Liste der Betroffenen Terminals
 - <times>: Liste der Zugriffszeiten
- Zeitformat: <Liste der Tage><Start-Zeit>-<End-Zeit>
- Tage: Mo,Tu,...,So,Wk,Wd,Al
- Beispiel:
sshd; *; !MoTu0800:1600

Ressourcenlimits: pam_limit

- Konfiguration: **/etc/security/limit.conf**
- Limitierung von:
 - allen Ressourcen, die ulimit beschränken kann
 - zusätzlich: maxlogins
- Soft & Hard Limits
- Syntax: **<domain> <soft|hard> <item> <value>**
- Beispiel:

foobar	soft	nproc	10
foobar	hard	nproc	20
- Priorität kann nur mit einem Hard Limit versehen werden

Herzlichen Dank für Ihr Interesse

Noch Fragen?

Emailadresse des Referenten:	fbrand@redhat.com
Homepage des Referenten:	people.redhat.com/fbrand
PAM Mailingliste:	pam-list@redhat.com